

# Organización del Computador 2

## Recuperatorio del segundo parcial

11/07/2019

Corregido: Facundo

### Normas generales

1 (30)	2 (50)	3 (20)	77 (A)
30	40	7	

- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Entregue esta hoja junto al examen, la misma no se incluye en la cantidad total de hojas entregadas.
- Está permitido tener los manuales y los apuntes con las listas de instrucciones en el examen. Está prohibido compartir manuales o apuntes entre alumnos durante el examen.
- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido y LU.
- La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los parciales tienen tres notas: I (Insuficiente): 0 a 59 pts, A- (Aprobado condicional): 60 a 64 pts y A (Aprobado): 65 a 100 pts. No se puede aprobar con A- ambos parciales. Los recuperatorios tienen dos notas: I: 0 a 64 pts y A: 65 a 100 pts.

### Ej. 1. (30 puntos)

- (20p) a. Considerando la siguiente tabla de traducciones de direcciones por segmentación y paginación. De ser posible, dar un conjunto de descriptores de segmento, directorio de paginas y tablas de paginas que cumplan con todas las traducciones simultáneamente. Detallar los campos de todas las estructuras involucradas. Además indicar si la traducción es *identity mapping* y en el caso de que alguna traducción no sea posible indicar por qué.

	Lógica	Lineal	Física	Características
a)	0x0013:0x00002FF0	0x00924FF0	0x00F00FF0	Datos Nivel 3 Acción: Lectura de 1KB, CPL:1
b)	0x0111:0x00123123	0x00123123	0xC0A00123	Código Nivel 1 Acción: Escritura de 4 Bytes, CPL:0
c)	0x0122:0x000FF202	0x09FFF202	0x00211202	Código Nivel 0 Acción: Ejecución de 3 Bytes, CPL:2
d)	0x0120:0x00000FFF	0x09F00FFF	0x00AAFFFF	Código Nivel 0 Acción: Lectura de 2 Bytes, CPL:0

- (10p) b. ¿Es posible activar paginación si la base del segmento actual es distinta de cero? Justificar. Además, de ser posible, proponer un conjunto de traducciones que respeten lo pedido, caso contrario explicar por qué no es posible.

### Ej. 2. (50 puntos)

Un sistema en dos niveles de protección con segmentación y paginación activa, ejecuta exactamente  $n$  tareas de forma concurrente. A cada tarea se le asocia otra, contabilizando un total de  $n$  pares de contextos de ejecución que deben ser administrados por el sistema.

Se proveen dos servicios:

- **cambiarTarea:** La tarea que llame a este servicio es intercambiada con la otra tarea en el par de tareas. La tarea que estaba ejecutando debe dejar de correr inmediatamente. El servicio no toma ningún parámetro.
- **dameRet:** Este servicio obtiene valores de la otra tarea en el par, justo antes que dejará de ser ejecutada. En EAX retorna el valor contenido en [EBP+4].

Considerar que cualquiera de las tareas puede cometer una excepción. En ese caso la tarea debe ser restaurada. Para esto se cuenta con la función `restaurarMemoriaDeLaTarea(cr3* memorymap)` que

1/2  
→ tengo que resetear todo (que corra desde el ppio).



*después de la excepción la tarea avanza de 0*

toma un puntero a un valor de 4 bytes que representa una estructura de CR3 y restaura toda la memoria de la tarea.

- (10p) a. Definir un posible mapa de memoria para las tareas. Indicar el rango de direcciones de las páginas utilizadas para el kernel y tareas. Indicar qué información específica es almacenada en cada rango designado (*Código/Datos, Page Directory, Page Table y Pila Nivel Cero*). Definir las entradas de la IDT para los servicios pedidos, para las rutinas de excepciones y para la interrupción de reloj. Además indicar el contenido de la TSS de una tarea al iniciar el sistema, explicando el contenido de cada uno de sus campos o los valores que deben tomar.
- (20p) b. Implementar en ASM y/o C la rutina de atención de interrupción del reloj y una rutina de las excepciones. Para ambas se debe escribir todo el código necesario para su implementación y declarar las estructuras de datos que hagan falta.
- (20p) c. Implementar en ASM y/o C la rutina de atención de interrupciones de los servicios del sistema.

Nota: Suponer que se cuenta con las definiciones de las estructuras del sistema para operar desde C.

### Ej. 3. (20 puntos)

Construir las siguientes funciones en C o en ASM:

- (10p) a. `uint32_t countUserInts(idt_descriptor* idtDesc)` toma un puntero a un descriptor de idt y cuenta la cantidad de rutinas de atención de interrupción que pueden ser llamadas desde nivel 2.
- (10p) b. `uint32_t countLevelZeroTasks(gdt_descriptor* gdtDesc)` toma un puntero a un descriptor de gdt y cuenta la cantidad de entradas de tss cuyo selector de segmento de código es de nivel 0.

```
typedef struct str_gdt_descriptor {
    uint16_t gdt_limit;
    uint32_t gdt_base;
} __attribute__((__packed__)) gdt_descriptor;
```

```
typedef struct str_gdt_entry {
    uint16_t limit_0_15;
    uint16_t base_0_15;
    uint8_t base_23_31;
    uint8_t type:4;
    uint8_t s:1;
    uint8_t dpl:2;
    uint8_t p:1;
    uint8_t limit_16_19:4;
    uint8_t avl:1;
    uint8_t l:1;
    uint8_t db:1;
    uint8_t g:1;
    uint8_t base_31_39;
} __attribute__((__packed__)) gdt_entry;
```

```
typedef struct str_idt_descriptor {
    uint16_t idt_limit;
    uint32_t idt_base;
} __attribute__((__packed__)) idt_descriptor;
```

```
typedef struct str_idt_entry {
    uint16_t offset_0_15;
    uint16_t segsel;
    uint8_t zero;
    uint8_t GateType:4;
    uint8_t s:1;
    uint8_t dpl:2;
    uint8_t p:1;
    uint16_t offset_16_31;
} __attribute__((__packed__)) idt_entry;
```

Suponer que se cuenta además con la estructura de tss.



① segmentación:a) 0x0013

OFFSET: 0x2FF0

DPL = 3

CPL = 1

RPL = 3

00010011

índice en  
GDT: 2RPL = 3  
GDT

base segmento + offset = dir. lineal, por lo tanto

$$\text{base segmento} = 0x924FF0 - 0x2FF0$$

$$= 0x922000 \quad \checkmark$$

b) NO se puede realizar esta traducción ya que los segmentos de código no permiten escrituras. Si se intentara realizar esta acción se generaría una #GP. ✓

c) 0x0122

OFF: 0xFF202

{ para poder ejecutar  
seguir segmento con

DPL = 0 teniendo

CPL = 2, este debe ser  
conforming necesariamente

000100100010

GDT RPL = 2

índice: 36 (0x24)

$$\text{base segmento} = 0x9FFF202 - 0xFF202$$

$$= 0x9F00000 \quad \checkmark$$

d) 0x0120

OFF: 0xFFF

DPL = 0

CPL = 0

RPL = 0

000100100000

RPL = 0  
GDT

índice: 36 (0x24)

$$\text{base segmento} = 0x9FOFFF - 0xFFF$$

$$= 0x9F00000 \quad \checkmark$$



$L = 0$   
 $AVL = 0$   
 $D/B = 1$   
 $S = 1$   
 $P = 1$

GDT: (para todas las entradas menos la nula):

#	base	G	límite	DPL	Tipo	descriptor nulo
nulo) 0	0	0	0	0	0	
(a) 2	0x0922000	1	0xFFFFF (4GB)	3	2	
c y d) 36	0x9F00000	1	0xFFFFF (4GB)	0	14 (conforming read/execute)	✓

Nota: Los segmentos de 4GB sin base nula tienen un espacio de direcciones inutilizadas

no se menciona en lím. de memo. física

Además de los segmentos desde donde se realizan las acciones: son 3 de código con DPL 0, 1 y 2. los tomo como flat (base 0 y límite 0xFFFF con G=1), de lectura y ejecución. (tipo 3)

paginación:

a) 0x00924FF0  
 ↓  
 0000 0000 1001 0001 0100  
 PD-off: 2 PT-off: 276

lectura de 1KB  
 se leerá de la dir. lineal  
 0x00924FF0 a 0x009253EF  
 (+ 1KB = 0x400B) - 1

por lo que hay que tener que mapear dos PTE.  
 (off 276 y 277) ✓

c) 0x9FFF202  
 ↓  
 0000 1001 1111 1111 1111  
 PD-off: 19 PT-off: 1023

ejecución de 3B  
 se lee de 0x9FFF202 a 0x9FFF204  
 (mapeado x una sola PTE) ✓

d) 0x9F00FF  
 ↓  
 0000 1001 1111 0000 0000  
 PD-off: 19 PT-off: 768

lectura de 2 Bytes.  
 ↓  
 se lee de 0x9F00FF a 0x9F01000  
 así que necesito mapear 2 páginas de memo.  
 (off 768 y 769) ✓



2/6

PD:

	#	dir base PT	U/S	R/W	P
(a)	2	&PT1 >>12	1 (user)	0	1
(c/d)	19	&PT2 >>12	0	0	1

los demás bits en 0.  
(PCD y PWT no se aclaran en el enunciado para PD ni PT)

PT1:

	#	dir base pag	U/S	R/W	P
(a)	276	0x00F00	1	0	1
	277	0x00F01	1	0	1

los demás bits en 0

PT2:

	#	dir base pag	U/S	R/W	P
(d)	768	0x00AFA	0	0	1
	769	0x00AAB	0	0	1
(c)	1023	0x00211	0	0	1

(privilegio 2 se mapea a supervisor en paginación)

Ninguna de las traducciones es identity mapping ya que dirección virtual y física que mapea nunca coinciden. ✓

(\*) El nivel de privilegio 2 de la segmentación se mapea a nivel usuario de paginación



(Asumiendo que esté prendido el bit PE, si no activa paginación causa #GP)

b) Paginación es la traducción de dirección lineal a dirección física, mientras que segmentación es la traducción de dirección lógica a dirección lineal. Si la base del segmento al que quiero acceder mediante una dirección lógica es distinta a 0, el único impacto que sentirá será paginación es que habrá direcciones lineales nunca usadas (suponiendo que ningún segmento empieza en 0) pero no modifica su funcionamiento. La base del segmento desde donde se intenta realizar el acceso a memoria <sup>(o mejor paginación)</sup> no tiene incidencia en paginación el funcionamiento de paginación.

Los accesos del punto a son ejemplos de esto, e incluso si alguna de las acciones (por ejemplo la d) se realizara del mismo segmento a donde se pide la información, paginación funcionaría sin problemas.

Nota: Faltó dar el conjunto de traducciones

De acuerdo al manual de Intel:

"... Intel (...) architectures do not enforce correspondence between the boundaries of pages and segments. A page can contain the end of one segment and the beginning of another..." Por lo que un segmento no alineado al tamaño de página tampoco causa problemas.

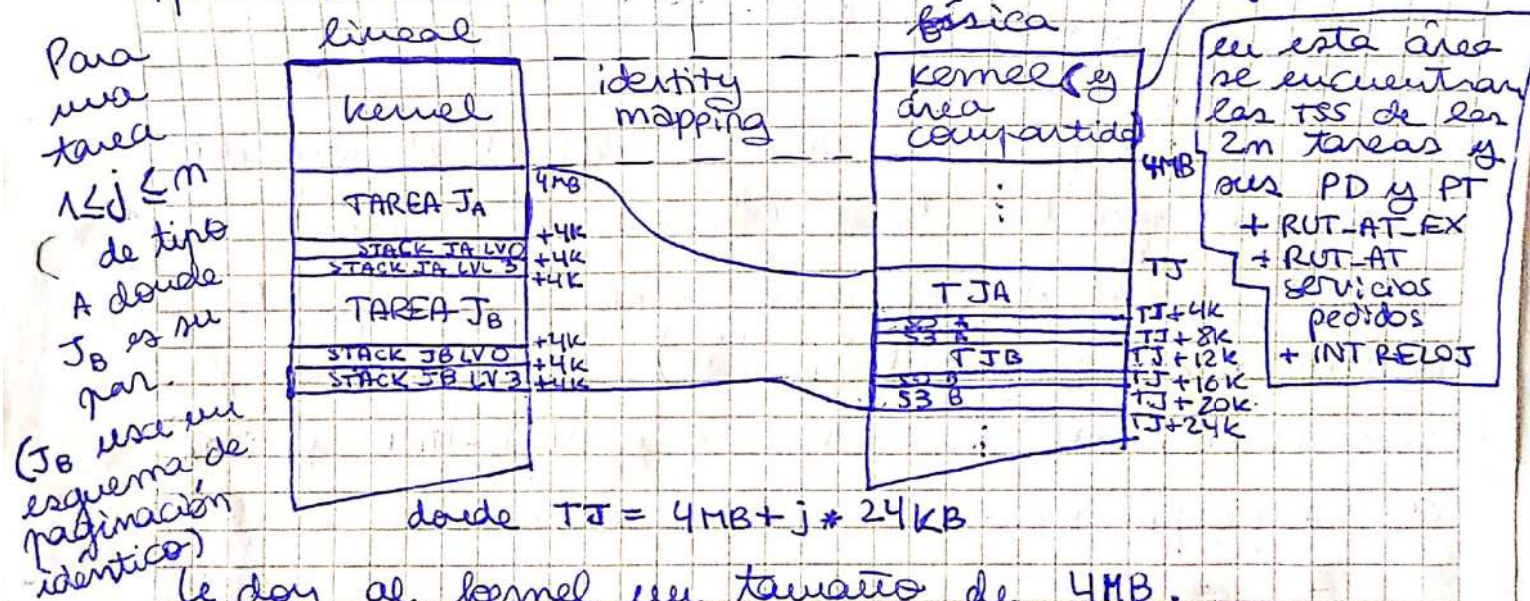


② Buen nivel USER = DPL 3, SUPERVISOR = DPL 0

en GDT. No se aclara ningún mecanismo para obtener pares de tareas así que <sup>de las dos siempre come.</sup> asumo que alguna

a) Utilizo segmentación flat. Tengo segmentos de código nivel 0 y 3 y de datos nivel 0 y 3. Además tengo los descriptors de TSS para las  $m$  tareas de manera que la tarea 1 se encuentre en el índice 5 y su par correspondiente, en  $5+m$  (es decir, los índices 5 a  $2m+4$  corresponden a tareas). Asumo que tengo suficientes entradas de GDT disponibles para las tareas.

# |  
1 | cod 0  
2 | cod 3  
3 | dat 0  
4 | dat 3  
(en 0 el seg. nulo)



le doy al kernel un tamaño de 4MB.

Como cuando solo TSS, PD y PT de todas las tareas (la TSS de cada tarea ocupa 68 B y PD+PT de una tarea toman  $3 * 4KB$ ) me entra un buen número de tareas en este espacio. Asumo que es suficiente.

(A muy grandes rangos, entran 170 pares de tareas en 4MB libres.)